Camelot - a database for climate model output

Meteorol. Appl. 7, 83–90 (2000)

Camelot – a database for climate model output

J M Gregory, S F B Tett and E L Hibling, *Hadley Centre for Climate Prediction and Research, UK Meteorological Office, Bracknell, Berks RG12 2SY, UK*

Camelot is a database system designed for output from the climate model run by the Hadley Centre of the UK Meteorological Office. It contains metadata for climate model output held in a tape archive and on the workstation system used for analysis. The system is required because of the large and rapidly growing volume of archive data and the great variety of uses of the climate model. Camelot enables the user to find out about what has been archived and to retrieve data selectively to the workstations with specifications given in terms of metadata rather than files, without recopying data already held online. The PV-WAVE analysis environment used on the workstation system also allows the user to access the data by reference to its metadata.

1. Introduction

Camelot is a database system designed for output from the climate model run at the Hadley Centre for Climate Prediction and Research, which is part of the UK Meteorological Office. Climate simulation is one application of the 'Unified Model' (UM) (Cullen, 1993), which is a software suite written at the Met. Office and also used, in different configurations, for numerical weather prediction, ocean modelling and other purposes. At the Meteorological Office, the UM is run on a Cray T3E with 880 DEC Alpha processors.

Camelot is a metadata database, the 'metadata' being information about the climate model output data. The metadata describes what data is available and where it is stored (see section 4 for details). The metadata is three or four orders of magnitude smaller than the data; the data itself does not reside in Camelot.

For numerical weather prediction, the UM comprises an atmospheric general circulation model (GCM) and a land-surface model, with prescribed sea-surface conditions. This configuration is sometimes also used for climate simulation, but more often the atmosphere and land-surface model are coupled either to a 'slab' ocean, which is an ocean layer 50 m deep with prescribed heat transport, or to a three-dimensional dynamical ocean model. The last arrangement, the coupled ocean–atmosphere GCM, is computationally the most demanding but is essential for prediction of time-dependent climate change. It is therefore heavily used and probably accounts for the majority of our Cray CPU time and data generated. The ocean GCM can also be run by itself, using prescribed atmospheric boundary conditions.

2. Climate model output

The 'climate model output' with which we are concerned consists of spatio-temporal fields of physical quantities simulated by the UM. As the model steps forward through time, it calculates time-dependent spatial fields of these quantities. Within the model, all fields are discretised on a latitude–longitude grid, possibly with a rotated pole. Both atmosphere and ocean GCMs use finite-difference techniques. For climate simulation, the usual spatial resolution of the atmosphere GCM is 3.75° of longitude by 2.5° of latitude, making a grid of 96×72 points to cover the globe. The latest ocean GCM has a resolution of 1.25° in both directions, giving six times as many points, on a grid of 288×144. Components of the climate model can also be run for limited areas, for instance the north Atlantic, Europe and the tropical Pacific, and at higher or lower resolution.

Some quantities are defined on a special level, such as the Earth's surface (e.g. surface pressure) or the top of the atmosphere (e.g. outgoing longwave radiation). Others (e.g. winds) have a third spatial dimension of model level number, corresponding to the vertical coordinate. In climate runs, the atmosphere has 19 and the ocean 20 irregularly spaced levels; there are also other sets of model levels, for instance the 4 levels of the soil temperature and hydrology model.

Quantities calculated within the model are of two types:

- *Prognostics*. These are the variables which determine the model evolution, being passed from one timestep to the next. There are about 250 of them, including (to name a few) surface pressure, temperature, horizontal winds and ocean currents, atmospheric specific humidity and seawater salinity.
- *Diagnostics*. These provide extra information which need not be preserved at the end of the timestep. Although they are in this sense optional, for any particular analysis some of them will be essential, for example radiative fluxes, cloud

amounts, precipitation, mixed-layer depth, vertical velocities. There are about 1 200 diagnostics available. Some diagnostics have a physical coordinate, such as pressure, rather than a model level number, as a third spatial dimension.

For a given configuration of the model, not all of the prognostics or diagnostics will be defined; for instance, ocean currents will not exist if the ocean GCM is not in use. For the coupled ocean–atmosphere GCM, the majority are available. Most quantities defined in a given configuration are available on every timestep. In the standard climate model, the timestep is 30 minutes in the atmosphere and 1 hour in the ocean.

Outputting all model quantities on all available timesteps, while permissible, would be undesirable because the vast quantity of output data would defy manipulation. For this reason, the UM contains a component known as STASH (Spatial and Temporal Averaging and Storage Handling), which reduces the output volume by various means:

- *Selection*. All quantities are optional on output. Most diagnostics are not computed at all if not required for output.
- *Temporal reduction*. Instead of being output on every timestep, quantities can be saved at intervals, for instance once a day. More commonly, they are aggregated over intervals, usually by averaging, but taking the maximum or minimum is also possible. In a typical climate run, most selected quantities are output only as means over all timesteps for a month and longer periods.
- *Spatial reduction*. Fields can be output with averaging over one, two or three spatial dimensions.
- *Spatio-temporal reduction*. It is possible to average a field over a specified spatial domain at each of a set of timesteps, and then assemble these numbers into a timeseries. An example would be a timeseries of the temperature averaged over the volume of an ocean basin at midnight on 1 September of successive years.

STASH is a component of the model itself, so these derived quantities are computed within the model. An alternative to this approach would be to write out the selected quantities at all timesteps to temporary storage, and to post-process them to obtain the required reduction. This would demand a great deal of temporary storage and an intricate post-processing setup.

Climate model output through STASH is written to the storage medium in 'PP' format, which is peculiar to the Meteorological Office. The only structure supported by PP format is a two-dimensional field. Quantities on several vertical levels are written out as vectors of two-dimensional fields, each with its vertical coordinate specified. Zonal and meridional means have a size of unity for one of their dimensions. Timeseries fields resulting from spatio-temporal reduction are also written out as two-dimensional fields, with time as one axis and spatial domain number as the other. Each PP field comprises two Fortran records: a 'header' record containing the field metadata (described below) and a data record. At standard climate resolution, a normal atmosphere field occupies around 28 kbyte, an ocean field 166 kbyte.

Fields are grouped into files under the control of STASH. Since they are written as the model runs, a natural division into files is by meaning period (e.g. fields for a given month). They are also separated according to model component (atmosphere or ocean) and further distribution into output streams can be specified by the user (e.g. timeseries fields may go to separate files). A typical file contains hundreds of fields and so has a size of 10–100 Mbyte. The UM writes the files to Cray disk as it runs, and a separate process, the UM server, copies them to disk on the IBM MVS system, which hosts the archive, also depositing control files on MVS disk (Figure 1). Another separate process, the archive server, running on MVS, copies the PP files to magnetic tape cartridges, following the instructions from the UM server in the control files. The archive server does not belong to any particular UM job; it continually checks for and acts upon control files from any job. Data compression takes place both in the UM software and in the tape drive hardware. The tapes are 10 Gbyte cartridges housed in a robotic tape archive.

3. Requirement for a database system

Climate model output is produced by climate experiments. Each experiment is the property of an individual user and is designed by that user for a specific purpose and uses a particular configuration of the model. The requirement for a database for climate model output is twofold. The first motivation derives from the flexibility of the UM system and STASH. As regards output, climate experiments differ in respect of:

- Availability and selection of quantities to be output.
- Temporal and spatial reduction.
- Organisation of output fields into output streams.
- Period of simulated dates of the run (for instance, from 1 September 1990 to 1 September 2100).

On account of all these differences, the user has a need to be able to find out what has been stored as output from a given experiment. We refer to this as the *query* function.

The other aspect of the problem is the volume of output. More than 70% of the Cray CPU time is used for climate simulation. A typical climate experiment may

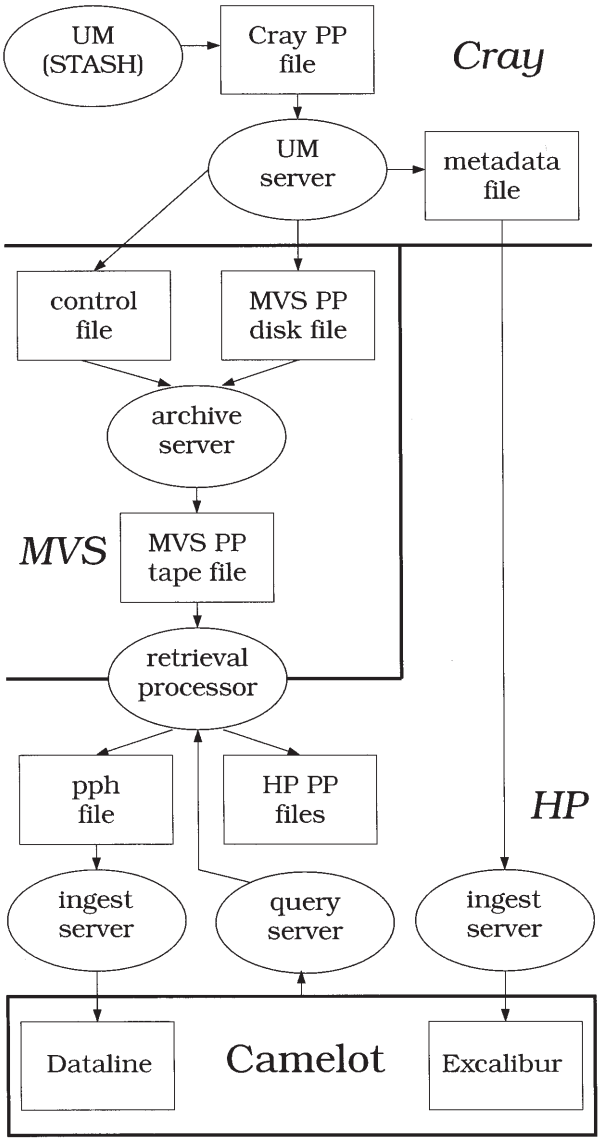

Figure 1. *Flow of model data and metadata. The model runs on the Cray. The IBM MVS system is the host for the data archive. The HP workstation system is used for interactive data analysis.*

run for a hundred years of simulated time and produce thousands of files. In the four years from September 1994, when the database was first set up, to the time of writing (September 1998), 2 200 experiments have produced between them 20 Tbyte of field data in 580 M fields (M = million) in 1.4 M files (Figure 2). (The figure for field data assumes 4 bytes for a floating-point word, the precision of the archive format, although the original Cray files have 8-byte words.) Moreover, we keep many experiments for a period of years, while the rate of production has been increasing and may be expected to continue to increase in line with supercomputer CPU power. Data analysis is undertaken on an HP-UX workstation system with a storage capacity on disk of about 500 Gbyte, which is only a small percentage of the total volume of data archived on tape. Hence there is a need for the ability to be selective in retrieving from the archive. To do this without excessive labour, the user should not have to know the contents or the names of the files in which the fields are stored. It should be possible to specify which fields are wanted in 'scientific' terms. We call this the *retrieval* function.

4. Design of the Excalibur database

Camelot is the overall database framework which has been set up to offer the user the query and retrieval functions. Within Camelot, Excalibur is the database of metadata describing the climate model output in the tape archive. Excalibur has been implemented as a relational database using software supplied by Empress. All database access is in standard query language (SQL) and so is in principle fairly portable.

The basic input to Excalibur is the metadata describing a single PP field. This has the following components:

- `source` and `filename`. The `source` is the name of the experiment, and the `filename` the name of the tape file containing the field.
- `posn`. The ordinal number of the field within the file (1, 2, 3, . . .).
- `stime` and `etime`. These specify the start and end of the period of simulated time to which the field applies. For example, a monthly mean for January 1861 has an `stime` of midnight on 1 January 1861 and an `etime` of midnight on 1 February 1861, one month later. For instantaneous fields, `stime` and `etime` are equal.
- `spatial`. This is a set of attributes describing the horizontal grid and vertical level of the field.
- `quantity`. Various codes are used to specify what the physical quantity is which the field contains (pressure, temperature, etc.).

For convenience, all times within the database are encoded as floating-point Julian day numbers. Double precision is needed for these, since times which differ by a second but span a range of 10^4 years might need to be distinguished, requiring a precision of about 13 decimal or 40 binary digits, more than can be represented in a four-byte floating-point or integer number. The choice of unit of time is not important, since it does not affect the precision. Years or seconds could equally well have been used.

In all, the metadata for a PP field amounts to 154 bytes per field, composed thus: 26 for `source` and `filename`, 18 for `posn`, `stime` and `etime`, 110 for `spatial` and `quantity`. One possible approach for building the database would be to put all this raw metadata in a single table, one row per field. With 580 M fields, the table would occupy 89 Gbyte, approaching a fifth of the total workstation storage, and probably being slow to search. This, therefore, is not a satisfac-

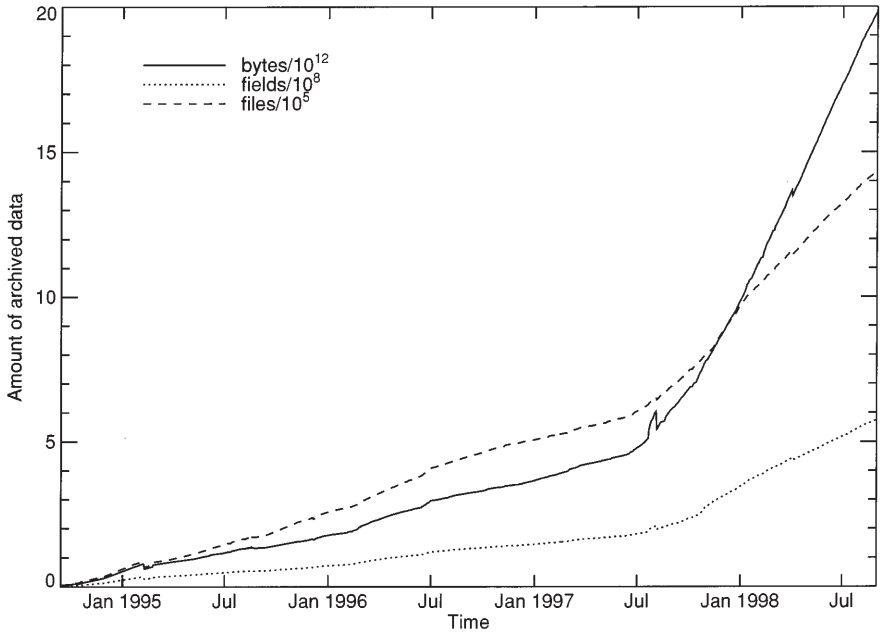

Figure 2. *Growth of archived data. The marked acceleration in data production in the middle of 1997 arose because of the switch-over from the less powerful Cray C90 to the Cray T3E.*

tory approach. We have to reduce the database size by eliminating redundant information. (Note that the figures used in this discussion are current at the time of writing. The database is constantly growing, but the arguments will remain the same.)

The first redundancy we exploit is that on the time-independent portion of the field description, the `spatial` and `quantity` information. There are relatively few distinct grids and levels used in the model, and only about a thousand different quantities. Hence, the number of different combinations of `spatial` and `quantity` attributes will be far fewer than 580 M. It is, in fact, about 60 000. If we assign a unique arbitrary long-integer index, which we call the `fieldid`, to each such combination, we can split the original table into two (a step of normalisation). One still has an entry for each of the 580 M fields, specifying for each its `fieldid`; the other gives the translations of `fieldid`s into `spatial` and `quantity` values. The latter is called the `field` table. This reduces the space taken by spatial and quantity information from

$$580\ \text{M} \times 110\ \text{byte} = 64\ \text{Gbyte}$$

to

$$580\ \text{M} \times 4\ \text{byte} + 60\,000 \times (110 + 4)\ \text{byte} = 2.3\ \text{Gbyte}$$

a factor of 27. Furthermore, following a second normalisation step described below, almost all the first term is eliminated, leaving us with just 6.8 Mbyte from the second; under these circumstances, the `fieldid` approach reduces the space required for this information by a factor of 9 300, and the size of Excalibur from 89 Gbyte to 25 Gbyte.

A possible second step would be to assign an index to the `source` and `filename` combination, since it is obviously redundant to repeat this for each of the hundreds of fields in a file. The space required for the information associated with these attributes would become six times smaller, falling from

$$580\ \text{M} \times 26\ \text{byte} = 15\ \text{Gbyte}$$

to

$$580\ \text{M} \times 4\ \text{byte} + 1.4\ \text{M} \times (26 + 4)\ \text{byte} = 2.4\ \text{Gbyte}$$

there being 1.4 M files. This strategy would reduce the database size to 13 Gbyte. While worthwhile, it is not as effective as another possibility we can exploit.

This depends on the repetitive nature of the contents of files that a given experiment produces. For example, the fictitious contents of the atmosphere monthly mean files for January and February 1861 from experiment `cb9ee` might be as shown in Figure 3. The important feature is that the contents of the files are exactly the same except that the dates in the second file are one month later. We capitalise on that by splitting the table into one depending on absolute time and one on relative time (Figure 4). The first is the `file` table, which gives a reference time, called the `ftime`, for each file. The second is the `contents` table, which supplies the contents of a file of this kind in a way which is independent of absolute time. In the `contents` table,

crf.sa.cb9eea.pm61jan:

posn	stime	etime	meaning of fieldid
1	0GMT 1.1.1861	0GMT 1.2.1861	surface pressure
2	0GMT 1.1.1861	0GMT 1.2.1861	x-component of wind on level 1
3	0GMT 1.1.1861	0GMT 1.2.1861	x-component of wind on level 2
		etc.	

crf.sa.cb9eea.pm61feb:

posn	stime	etime	meaning of fieldid
1	0GMT 1.2.1861	0GMT 1.3.1861	surface pressure
2	0GMT 1.2.1861	0GMT 1.3.1861	x-component of wind on level 1
3	0GMT 1.2.1861	0GMT 1.3.1861	x-component of wind on level 2
		etc.	

Figure 3. *Example contents of two atmospheric monthly mean files for the same experiment.*

file table:

filename	ftime
crf.sa.cb9eea.pm61jan	0GMT 1.1.1861
crf.sa.cb9eea.pm61feb	0GMT 1.2.1861

contents table:

posn	rtime	ltime	meaning of fieldid
1	0 d	30 d	surface pressure
2	0 d	30 d	x-component of wind on level 1
3	0 d	30 d	x-component of wind on level 2
		etc.	

Figure 4. *Example of the* file *and* contents *tables.*

rtime gives the start of the meaning period, in days, relative to the ftime of the file, and ltime gives the end of the meaning period, in days, relative to the start time.

$$\texttt{stime} = \texttt{ftime} + \texttt{rtime}$$

$$\texttt{etime} = \texttt{ftime} + \texttt{rtime} + \texttt{ltime}$$

For an instantaneous field, ltime is zero. In this example, the ltime of 30 days indicates one month, as all months have 30 days in the climate model. (The normal calendar can also be handled.) Although in this instance rtime and ltime are the same for all fields in the file, this is not necessarily the case.

The file and contents tables are joined by assigning a unique contentsid for each different 'kind' of file, in terms of its contents list. The contents lists of all kinds of file are concatenated together in the contents table. The 1.4 M files can be classified as 11 000 kinds, and the contents table has 7.1 M rows, an average of 620 fields per contentsid. This normalisation step reduces the information for the source, filename, posn, stime, etime and fieldid by a factor of 120, from

$$580\ \text{M} \times (26 + 18 + 4)\ \text{byte} = 28\ \text{Gbyte}$$

to

$$1.4\ \text{M} \times (26 + 8 + 4)\ \text{byte} + 7.1\ \text{M} \times (4 + 18 + 4)\ \text{byte} = 240\ \text{Mbyte}.$$

(The additional 8 bytes in the rows of the file table are for the double-precision ftime attribute.) Since the first step of normalisation compressed the field metadata into 6.8 Mbyte, all the information from the original 89 Gbyte of raw input metadata is contained in about 0.25 Gbyte after the second step, a reduction in space by a factor of 350.

The design is completed (Figure 5) by a final step, whereby we assign a sourceid to each source (experiment), more for convenience than to save space. The source table also contains some information not shown in the figure describing the experiment, in particular the identity of the owner and a brief indication of the purpose; this metadata is also provided automatically by the UM to Excalibur. With the inclusion of cross-reference tables and indices, Excalibur expands to 850 Mbyte, still easily manageable on the workstation system.

5. Excalibur database functions

Excalibur is involved in three kinds of function: ingestion, query and retrieval (Figure 1). Ingestion is the process whereby it acquires the metadata for new UM files as they are produced. To accomplish this, the UM server writes a metadata file, which is plain text, to Cray disk at the same time as the PP file is generated. The metadata file is fetched to the workstation system by the Excalibur ingest server process and processed into the database structure we have described. Part of this task is to determine whether the new file is an instance of an existing type of file already described in the contents table, which is a rather awkward and slow operation in SQL. An alternative method has therefore been implemented using Unix filters and a program in Perl. The remainder of the ingestion is done in SQL. Using this scheme, the average time to ingest a file is only 4.8 s. At this rate, 22 000 files could be processed per day, which gives a very comfortable margin over the current rate of production of about 1 700 per day, enabling the server to catch up quickly should a backlog accumulate because of any system failures.

The query function enables the user to ask questions such as:

- Is daily mean soil moisture available from experiment cbtph?

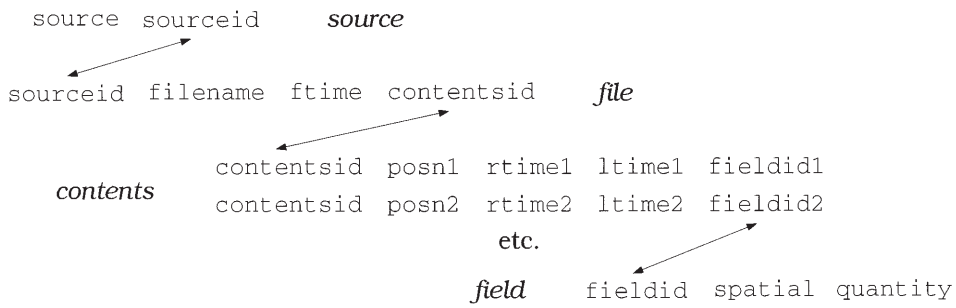

Figure 5. *Design of the Excalibur database. The labels in italic are names of tables, and those in* `typewriter` *style are the names of attributes (columns) of the tables. Double-headed arrows link attributes which are used to join tables.*

- What period of simulated years is covered by experiment `cblzs`?
- On what levels is monthly mean salinity archived in experiment `cbqmk`?
- What experiments are owned by user `jmgregory`?

Enquiries are formulated at the Unix command line using a shell script which generates and submits SQL for execution by the Camelot query server process and returns the output to the user. Such queries take tens of seconds to complete.

The retrieval function permits the user to request that selected fields should be fetched from tape in such terms as *January mean precipitation for 1980–1989 from* `cbt3a`. While not actually framed in English, the requests are given in 'physical' terms equivalent to these, and not with reference to tape files. Effectively, the user is specifying conditions on the field metadata. The retrieval tool interrogates the database through the query server, translating the request into a list of `filename`s and `posn`s within those files. The lists are used to construct a job which is submitted for execution by the MVS batch system. According to the volume of data and priority of the job, the fields are delivered to the workstation system in minutes to days.

Users can also fetch data from tape without using Excalibur, by specifying filenames and metadata requirements. This is often done when only a small number of files is concerned, especially if the intention is to fetch the entire contents of the files. The great majority of retrievals which have to read data from a large number of files are carried out using Excalibur, because it is much more convenient than constructing lists of dataset names.

6. Online data and the Dataline database

The final part of the system to be described concerns the form in which the data is stored on the HP-UX workstation system. This arrangement was not actually developed for Camelot but fits into the framework readily. Once again, PP fields and files are used, as in the archive. However, the data on the workstations is an ad hoc, transient, selection, and it is not appropriate to organise it in the same way as for archive. The fields, as they arrive, are divided among small files containing (with one kind of exception) a single field each, and having file basenames which indicate a good deal of the field metadata; for instance, annual mean surface air temperature for 1970 would be stored in a file

```
000100000000.03.236.00128.1970.09.01.00.00.pp
```

Not all the metadata is used but only a sufficient amount to distinguish the files which might be needed. (Other filenaming conventions can be used instead if the default is not adequate.) These rather long names are not user-friendly, clearly. The original purpose of the scheme was to enable analysis tools to find out what data existed in a directory by listing its contents, in effect using the directory file as a simple metadata database. The main disadvantage of this method is that the file basenames, long though they are, do not contain all the useful metadata.

Subsequently, an alternative method was devised, whose aim is to encourage the user to view the directory as a 'data-space' containing fields described scientifically rather than as a set of named files. In each directory, a file named `pph` is created, comprising just the headers of all the PP fields in the files in the directory (i.e. the metadata records of the fields). The PV-WAVE analysis software reads the `pph` file into memory when the user indicates the directory is to be used. There are commands available in the PV-WAVE software for query and retrieval, analogous to the Excalibur functions, which refer to the directory contents held in memory. For instance

```
stashlist,cblzs,ss(period=1)
```

lists all the quantities available as annual means (meaning period of one year) from the directory `cblzs` (*query* function), while

```
z=ppa(cblzs,ss(stash=3236,period=1,
        proc=128,yr=1970))
```

fetches the annual (`period=1`) mean (`proc=128`) field for 1970 for surface air temperature (`stash=3236`) into PV-WAVE variable `z` (*retrieval* function). In order to obtain the field, the `ppa` function first identifies the metadata record which satisfies the criterion, and then deduces from this the name of the file in which the field is stored. As the metadata is all held in memory, this procedure is faster than listing the directory itself. Furthermore, forward translation from metadata to filename is easier to implement than the reverse. The method does not depend upon any particular organisation of fields into files, and in future we may keep the fields in the larger groups in which they are typically retrieved from MVS. These may be assigned arbitrary names, so long as the analysis software has the information necessary to translate a metadata record to a filename. Use of smaller numbers of files would probably lead to more efficient I/O.

The metadata of the fields held on the workstation system is incorporated into a second database of Camelot, named Dataline. The ingest process for Dataline also processes the `pph` files, which are compiled as the fields are delivered from MVS (Figure 1). When asked to fetch fields from the archive, the Camelot retrieval processor identifies those which do not already exist online, by referring to Dataline, and retrieves only this subset. This saves workstation disk space and processing costs on MVS. It is particularly convenient for experiments which are in progress; the same request for a set of fields needed for monitoring the experiment can be submitted regularly (for instance, daily), and only those fields which have been produced since the previous such request will be fetched.

The Dataline database has a different relational model from Excalibur (Figure 6). The reason for this is that, as remarked above, the set of fields online is typically a rather ad hoc selection. In particular, it does not have the repetitive file contents structure of the archive. Because of this, the `contentsid` normalisation of Excalibur cannot be used for Dataline. The normalisations on `fieldid` and `sourceid` can be applied, and in fact the `field` and `source` tables are shared with Excalibur. The `contents` table of Dataline contains a row (46 bytes, which includes some attributes not shown Figure 6) for each individual field. This inefficiency, compared with Excalibur, does not present a problem because only about 10 M fields in total could be accommodated on the workstation system, implying a maximum size for Dataline of around 500 Mbyte. Currently, the Dataline `contents` table has 2 M rows. When online data is deleted, whether manually by users or by disk space management processes, its metadata is automatically removed from Dataline.

The PV-WAVE software can interrogate Dataline in order to locate online directories containing data which the user wants to access. This makes it convenient for users to share online data and to spread data over various filesystems. There is currently no link between PV-WAVE and the Excalibur retrieval function. That is, if the user discovers that the required data is not online, a request must be submitted manually to fetch it.

7. Conclusions

We have described the implementation of Camelot, a database system containing two relational databases, namely Excalibur and Dataline. Excalibur contains metadata describing fields archived by the climate model as they are produced, and Dataline the metadata of fields held on the workstation system used for analysis. Camelot offers tools for finding out what data exists in the archive, which is already large and growing fast, and retrieving it selectively to the workstation system, without unnecessary duplication in the online store. Both functions are performed by specifying conditions relating to the field metadata rather than the files containing the fields. Analogous query and retrieval tools, based on metadata, are also available for the online store in the PV-WAVE analysis software used on the workstation system.

An important principle in the design is that there are no assumptions about filenames or the existence of files, either in the tape archive or online. The system treats filenames as arbitrary metadata, and each file in the tape archive is individually processed and recorded. This approach has significant advantages. Firstly, the database can accommodate the absence of a random selection of files from an experiment, as can arise if a problem is encountered in running the experiment or if a tape breaks with loss of data. If the system assumed it could generate lists of filenames systematically, such problems would be much more awkward to handle.

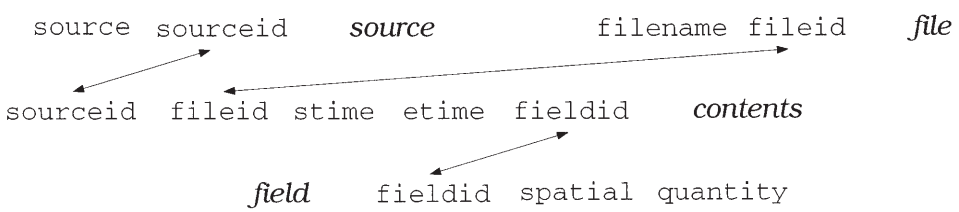

Figure 6. *Design of the Dataline database. The same conventions apply as in Figure 5.*

Secondly, changes of naming conventions for tape or workstation files do not present any difficulty for the database. (The PV-WAVE software does make use of online filenaming conventions, as we have seen.)

The design of the database depends on certain characteristics of the data generated by a typical UM experiment. Crucial among the assumptions is that the output will comprise many instances of a rather small number of different types of file. If we could not exploit this to make the `contents` normalisation, it is doubtful that Excalibur could be accommodated on the workstation system or that its functions would be fast enough. Equally important is the assumption that there are relatively few different types of field; Dataline depends on this as well as Excalibur. This holds because the UM offers a discrete and finite set of quantities which could be output, and experiments use only a rather small set of different horizontal grids and vertical levels. This is the case because changing the spatial resolution is a non-trivial task involving readjusting the scientific schemes of the model and deriving new ancillary input data. The design of Dataline further assumes that disk space limitations will restrict the number of fields held online. This assumption is simply a corollary of the need for selective retrieval, which is one of the motivations for implementing the database system.

The Camelot system was released for general use in the Hadley Centre at the end of 1995. In the course of the last three years, various modifications have been made to improve efficiency, but the essential design has not had to be changed and the performance has remained good, while the metadata volume has grown by an order of magnitude. As a result of user experience, improvements are planned in the interfaces, and we hope to expand the system by including processed data derived from climate model output.

Gridded observational data could also be included. Up to now, this has not been necessary as the observational datasets in use at the Hadley Centre are fairly small in number and each typically has a limited range of applications, so that it is not hard for users to remember which one they want to use and where to find it. However, as greater and more systematic use is made of datasets from operational reanalyses, a need is emerging to include these in Excalibur, because they have similar characteristics to UM data: they are voluminous, so selective retrieval is needed, and they contain a large range of different quantities, which may vary from time to time. If information from a variety of sources other than the UM were to be incorporated, the designation of the source by the UM experiment name alone (which is a five-character string chosen arbitrarily by the UM user interface) would not be sufficient. We would need to supplement the `source` table with detailed descriptions of the origin and purpose of the data. In a more general environment, it would become advantageous for this higher-level description of the source to be organised systematically so that it could be searched using database tools.

Should we move away from PP format for archiving, for instance to netCDF, alterations to the database would obviously be required, but this would not be very difficult as the overall structure would not be greatly affected; the metadata would carry essentially the same information, regardless of the file format. Indeed, the central idea of the software described here is its aim of providing an access method to data based not on files, but on metadata, offering a conceptual model that is more helpful to users of the data.

Acknowledgements

Many individuals at the Met. Office have been involved in the development of the UM and its archiving system. We are particularly grateful to Mick Carter for his encouragement of this project. We would like to thank Jan Polcher and Bob Drach for their helpful reviews of this paper. Our work was supported by the UK Department of the Environment, Transport and the Regions under contract PECD 7/12/37 and by the Public Meteorological Service Research and Development Programme.

Reference

Cullen, M. P. J. (1993). The unified forecast/climate model. *Meteorol. Mag.*, **122:** 81–94.